Software Qual J (2017) 25:3–5
DOI 10.1007/s11219-017-9354-x

Special issue on program debugging

Sudipto Ghosh[1] · **J. Jenny Li**[2]

Published online: 13 January 2017

Software today is large and complex, in fact, more so than ever before. Consequently, debugging when failure is observed is also becoming much more difficult and time-consuming. Manual debugging is quickly losing its viability as a practical option, and yet at the same time, various alternative approaches are still too immature for practical use.

Techniques that aim for automatic fault localization are not accurate and consistent enough to pinpoint the locations of faults to a desired degree. Distinguishing executions that fail due to different causative faults, reliably recording and replaying failed executions, and fixing bugs without introducing new faults are but some of the debugging-related problems faced by developers today.

Recent efforts, such as the recommender system-based approaches that mine different types of software repositories and suggest various debugging actions or program fixes, are still unproven to be consistently effective. Formal verification techniques generally suffer from complexity and scalability issues. Static techniques can often be imprecise. The heavy performance overhead of dynamic techniques can prohibit their application.

Even though a software development project may apply such techniques, they may require developers to make the final selection, diagnosis, and fixing decisions. Social aspects of software development projects that aid debugging, such as selecting the right developers to perform the right debugging tasks at the right time, have not been adequately explored. Last but not least, while studies are being conducted to reveal, clarify, or resolve some of these issues, researchers often conduct studies in restrictive environments that may inherently make incorrect assumptions about the industry. All these concerns can induce

✉ Sudipto Ghosh
ghosh@cs.colostate.edu

J. Jenny Li
juli@kean.edu

[1] Department of Computer Science, Colorado State University, Fort Collins, CO, USA

[2] Department of Computer Science, Kean University, Union, NJ, USA

in practitioners a lack of faith with regard to the results that debugging research can offer and deliver. In this special issue on Program Debugging, seven papers are presented that investigate issues in diverse areas of program debugging and their automation.

The first paper "Efficient and Scalable Omniscient Debugging for Model Transformation" by Jonathan Corley, Brian P. Eddy, Eugene Syriani, and Jeff Gray proposes a technique for supporting omniscient debugging for model transformations. By using enhanced navigation and exploration features during a debugging session, developers can achieve more thorough omniscient debugging than by using a strict stepwise execution environment. This paper also presents the results on comparing the execution time performance and memory usage of this approach compared to a stepwise execution approach.

The second paper "10 Years of Research on Debugging Concurrent and Multicore Software: A Systematic Mapping Study" by Sara Abbaspour, Daniel Sundmark, Sigrid Eldh, Hans Hansson, and Wasif Afzal presents the results of a systematic mapping study of debugging concurrent and multicore software. The findings indicate that several important issues need to be investigated and broader studies need to be performed.

The third paper "Studying the Advancement in Debugging Practice of Professional Software Developers" by Michael Perscheid, Robert Hirschfeld, Benjamin Siegmund, and Marcel Taeumel presents results obtained by observing several professional developers when they performed debugging tasks. Interviews were conducted and a large-scale online debugging survey was performed to obtain new insights on debugging practices.

The fourth paper "Reproducing Failures based on Semi-Formal Failure Scenario Descriptions" by Gun Karagoz and Hasan Sozer presents an approach that uses semi-structured failure scenario descriptions from the field to let developers automatically reproduce failures for debugging purposes.

The fifth paper "Effective Software Fault Localization using Predicted Execution Results" by Eric Wong, Ruizhi Gao, Zhenyu Chen, and Yabin Wang addresses the problem of failure detection when no test oracle exists to automatically determine the success or failure of executions. Often the outputs are verified manually and sometimes even the expected outputs are unknown.

The sixth paper "Fault Localization for Automated Program Repair: Effectiveness, Performance, and Repair Correctness" by Fatmah Yosef Assiri and James M. Bieman studies various fault localization and automated program repair techniques in terms of their effectiveness, performance, and correctness of the produced repairs.

The seventh paper "A Declarative Framework for Stateful Analysis of Execution Traces" by Naser Ezzati Jivan, Florian Wininger, and Michel R. Dagenais proposes a generic declarative trace analysis framework to analyze, comprehend, and visualize execution traces. This is an important step before developers can debug execution, detect problems and bottlenecks, and identify root causes for problems.

We, the guest editors, thank the authors for their hard work in preparing and revising their manuscripts. We thank the reviewers for taking the time to write detailed reviews. Finally, we thank the Editor-In-Chief, Prof. Rachel Harrison, and the editorial staff for their patience and hard work in getting this special issue ready for publication.

Dr. Sudipto Ghosh is an Associate Professor of Computer Science at Colorado State University. He received the Ph.D. degree from Purdue University in 2000. His teaching and research interests include modeling, designing and testing of object-oriented software and aspect-oriented and component-based software development. He is on the editorial boards of the Software Quality Journal, Journal of Software Testing, Verification and Reliability, and Information and Software Technology. He was a general co-chair of MODELS 2009 and Modularity 2015. He was a program co-chair of ICST 2010. He is a member of the ACM and a Senior Member of the IEEE, and IEEE Computer Society.

Dr. J. Jenny Li is a professor at Kean University computer science department. Prior to joining Kean, she was a research scientist at Avaya Labs, formerly part of Bell Labs. She is an experienced academic and industrial researcher with more than 80 papers published in technical journals and conferences, and holder of over 20 patents. She also worked at Bellcore (formerly Telcordia and now Applied Communication Science) for 5 years. Her current research interest is in the application of artificial intelligence and machine learning techniques to network software security. She received her Ph.D from University of Waterloo in 1996.